

稀疏矩阵与稀疏求解器

管唯宇

目录

- 稀疏矩阵
- 稀疏求解器
 - 稀疏矩阵与图论
 - 填充元与优化
 - 直接快速解法
 - STAPpp中的实现

稀疏矩阵的储存方式

- COO
- DIA
- CSR
 - 稀疏向量储存
- CSC
- SKYLINE
- ELL
- BSR
- ...

Coordinate (COO)

Diagonal (DIA)

Compressed Sparse Row (CSR)

ELLPACK (ELL)

①	⑦	0	0
0	②	⑧	0
5	0	3	9
0	6	0	4

Matrix

entries per row

0	1	*
1	2	*
0	1	2
1	3	*

column indices

padding

1	7	*
2	8	*
5	3	9
6	4	*

values

rows

稀疏求解器

- 求解步骤
 - 预处理：填充元优化
 - 符号分解
 - 数值分解
 - 回代和迭代优化
 - (内存释放)

稀疏求解器 填充元与优化

• 填充元 (FILL IN)

- 在矩阵LU分解过程中，零元位置在分解后出现了非零元
- 变换矩阵可以大幅消除填充元
- SKYLINE储存方式不存在填充元的问题

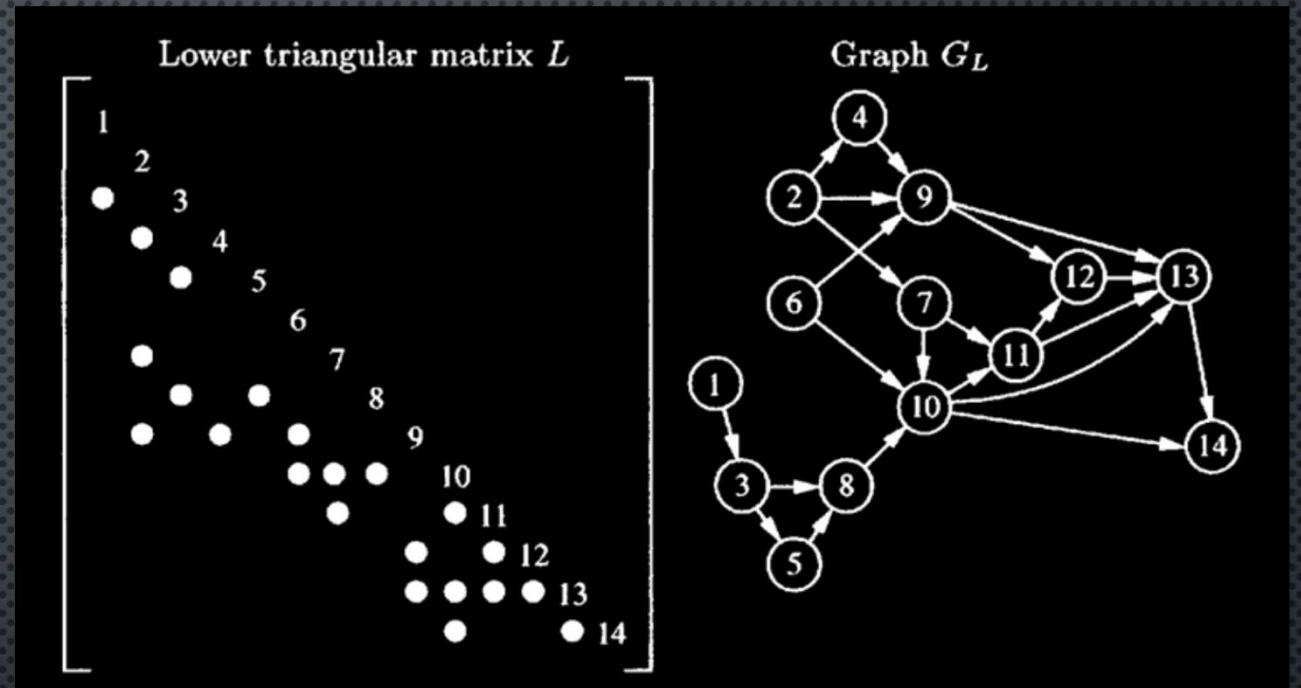
$$\mathbf{A} = \begin{pmatrix} 0.494095 & 0.157562 & 0.272554 & 0.13301 \\ 0.157562 & 0.505746 & 0. & 0.272554 \\ 0.272554 & 0. & 0.505746 & 0.157562 \\ 0.13301 & 0.272554 & 0.157562 & 0.494095 \end{pmatrix}$$

$$\mathbf{LU}^T = \begin{pmatrix} 0.4449095 & 0.157962 & 0.272554 & 0.13301 \\ 0.109329 & 103.585746 - 0.0297979 & 0.272554 & 0. \\ 0.189119 & -0.0220228 & 0.4345314 & 0.10254828 \\ 0.0922925970 & 0.1538965 - 0.0561828 & 0.236728 & \end{pmatrix}$$

稀疏求解器

稀疏矩阵与图论

- 矩阵对应的有向图 G
 - 对称矩阵退化为无向图
 - 最小度量法**
 - 挑选图 G 中的最小度量点，把它标定当前顺序号。
 - 多重剖分法**
 - 寻找图 G 中的一个子集 S ，使得去除这些点后，剩下的点被分割成几个互不相连的分块



定理 1. $L(i, j)$ 是填充元，当且仅当在矩阵 A 的对应图 G 中存在一个连接节点 i 和 j 的路径，路径上所有的点编号都不超过 $\min(i, j)$ 。

快速直接解法与消去树

消去树是一种数，其定义为 [2]

$$\text{Parent}[j] = \min \{i > j \mid l_{ij} \neq 0\}$$

关于消去树的性质，有如下定理 [2]:

定理 2. 如果 $l_{ij} \neq 0$ ，则 x_i 在消去树中是 x_j 中的祖先节点。

推论 1. 如果 $T[x_i]$ 和 $T[x_j]$ 是消除树中二独立的子数，则有，

$$\forall x_s \in T[x_i], \forall x_t \in T[x_j], l_{st} = 0$$

定理 3. 对于 $i > j$ ，则 l_{ij} 当且仅当在图 G 中存在一个路径

$$x_i, x_{p_1}, \dots, x_{p_t}, x_j$$

使得 $\{x_{p_1}, \dots, x_{p_t}, x_j\} \subseteq T[x_j]$

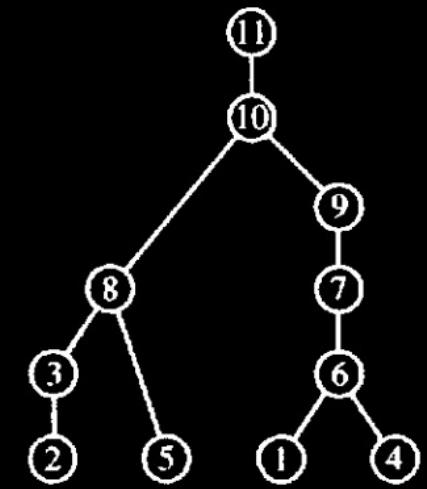
定理 4. $l_{ij} \neq 0$ 当且仅当

$$\exists x_k \subset T[x_i], \text{s.t. } a_{ik} \neq 0$$

Cholesky factor L of A

$$\begin{bmatrix} 1 & & & & & & & & & & \\ 2 & 3 & & & & & & & & & \\ & & 4 & & & & & & & & \\ & & & 5 & & & & & & & \\ & & & & 6 & 7 & & & & & \\ & & & & & & 8 & & & & \\ & & & & & & & 9 & & & \\ & & & & & & & & 10 & & \\ & & & & & & & & & 11 & \\ & & & & & & & & & & \end{bmatrix}$$

elimination tree of A



STAP++中程序实现

CSRMatrix: SparseMatrix

```
CSRMatrix::size  
CSRMatrix::values  
CSRMatrix::columns  
CSRMatrix::rowIndexes  
CSRMatrix::_tempColumns std::set<int>  
CSRMatrix::beginPostionMark()  
CSRMatrix::markPosition(int, int)
```

使用 std::set 的好处

插入速度快，灵活

避免内存碎片化，由 STL 管理内存

CSRSolver: Csolver

```
CSRSolver::CSRSolver(CSRMatrix&)  
CSRSolver::solve(double*, unsigned NLCASE)
```

Thanks